XML

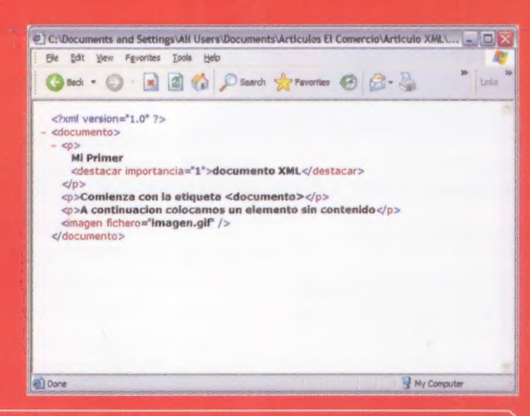
EXTENSIBLE MARKUP LANGUAGE

PC WORLD PERÚ

XML es un metalenguaje desarrollado con la idea de aprovechar las ventajas del HTML, pero que, a su vez, corrija las limitaciones de este.

Código en HTML

```
<p>
<dt>
<b>
<a
href="/exec/obidos/ASIN/0764531999/qid=
919015337">
Xml : Lenguaje Extendido de
Marcas</a></b> ~
<NOBR><font color=#990033>Aprenda en
24 horas</font></NOBR>
<dd> Elliotte Rusty Harold / Paperback /
Publicacion1998
<br>
Precio: $31.99 ~
<NOBR><font color =#990033>Descuento:
$8.00 (20%)</font></NOBR>
<br>
<a
href="/exec/obidos/ASIN/0764531999/qid=
919015337">
<i>Read more about this title...</i></a>
```



A la izquierda, la programación en HTML; y a la derecha, en XML, para la venta del siguiente libro en una tienda virtual:
XML: Lenguaje Extendido de Marcas ~ Aprenda en 24 Horas
Elliotte Rusty Harold / Paperback / Publicacion1998
Precio: $31.99 ~ Descuento: $8.00 (20%)
Leer mas información acerca de este titulo...

Código en XML

```
<?xml version="1.0"?>
<libro>
  <titulo>Xml: Lenguaje Extendido de
Marcasge</titulo>
  <disponible tiempo="24"
unidad="hours"/>
  <autor>Elliotte Rusty Harold</autor>
  <formato>Paperback</formato>
  <publicacion>1998</publicacion>
  <precio cantidad="31.99"
moneda="dolar"/>
  <descuento cantidad="20"/>
  <enlacelibro
href="/exec/obidos/ASIN/0764531999/qid=
919015337"/>
</libro>
```

LINEAMIENTOS GENERALES

XML permite trabajar de forma más eficiente. XML describe una clase de objetos de datos llamados documentos XML, así como el comportamiento de los programas de computadora que los procesan.

Los documentos XML están compuestos por unidades de almacenamiento llamadas entidades, que contienen tanto datos analizados como no analizados. Los datos analizados están compuestos de caracteres, algunos de los cuales corresponden a la forma de datos caracter, y otros a la forma marca. Las marcas codifican una descripción de la estructura de almacenamiento del documento y su estructura lógica. XML proporciona un mecanismo para imponer restricciones al almacenamiento y a la estructura lógica.

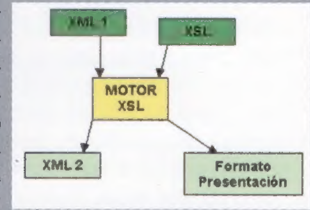
Se utiliza un módulo software llamado procesador XML para leer documentos XML y proporcionar acceso a su contenido y estructura. Se asume que un procesador XML hace su trabajo dentro de otro módulo, llamado aplicación. Esta especificación describe el comportamiento requerido de un procesador XML en términos de cómo leer datos XML, y la información que debe proporcionar a la aplicación.

El lenguaje HTML, a pesar de su sencillez, es sin duda un invento prodigioso. Pero a su vez, el HTML ha sido víctima de su propio éxito. El gran crecimiento de Internet, los intereses comerciales y la necesidad de poder realizar páginas web vistosas, ha dado lugar a que en poco tiempo este lenguaje haya evolucionado muy rápidamente y, por desgracia, no siempre por el camino más adecuado. Sigue siendo igual de rígido e inflexible como era en un principio.

Estas razones han obligado a los miembros del W3 Consortium a desarrollar un nuevo lenguaje (mejor dicho, metalenguaje) denominado XML (Extensible Markup Language) que aproveche las innegables ventajas del HTML pero que a su vez permita realizar muchas cosas más. Esto no significa, al menos por el momento, el fin del HTML. Existen demasiadas páginas en HTML, y resulta muy sencillo crearlas. Además los navegadores no soportan todavía XML en toda su potencia.

La idea que subyace bajo XML es la de crear un lenguaje muy general que sirva para muchas cosas. El HTML está diseñado para presentar información directamente a los humanos, pero es un lenguaje complicado de procesar. El HTML no indica lo que está representando, mientras que XML describe el contenido de lo que etiqueta.

Esto permite, por ejemplo, realizar motores de búsqueda mucho más eficaces para un acceso más rápido y eficiente a la información. La potencia de esta forma de trabajar radica en que se está etiquetando e identificando el contenido, sin tener en cuenta la forma de presentarlo. El W3C está trabajando actualmente en el desarrollo de un lenguaje de hojas de estilo denominado XSL (Extensible Style Languaje) que permita generar el tipo de presentación o salida de la información.



Si el HTML supuso una revolución porque permite la comunicación entre las personas, el XML supondrá una revolución porque va a permitir la comunicación entre las máquinas.

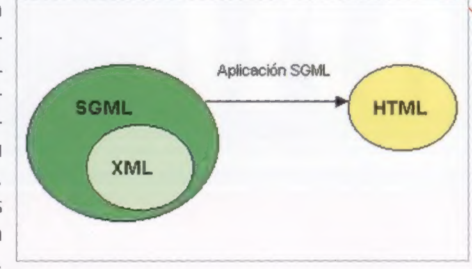
▸▸▸ HTML, XML, VERSUS SGML

▸ XML no es un HTML++. Tanto el XML como el HTML tienen su base en el SGML. El SGML (Standard Generalized Markup Language, ISO 8879) es el estándar internacional para la definición de la estructura y el contenido de diferentes tipos de documentos electrónicos. Es decir, es un metalenguaje que permite definir lenguajes, que a su vez, permiten definir la estructura y el contenido los documentos. La definición de la estructura y el contenido de un tipo de documento se realiza a través de una DTD. En ella se definen los elementos que conformarán ese tipo de documento, y como tienen que estar organizados para que sea correcto.

▸ Por ejemplo, en una DTD se define cómo deben ser los documentos HTML. Por tanto, el HTML no es más que un tipo de documento SGML que se utiliza en la web; y esto es importante, ya que aquí radica su principal diferencia con el XML.

▸ El XML no es ningún tipo de documento SGML, sino que es una versión abreviada de SGML optimizada para su utilización en Internet. Esto significa que con él se pueden definir los diferentes tipos de documentos y etiquetas, y, por tanto, ya no se dependerá de un único e inflexible tipo de documento HTML.

▸ El XML, más que un HTML++, hay que considerarlo como un SGML optimizado para su utilización en Internet. Como escribió Richard Ligth en su libro Presenting XML, "XML ofrece el 80% de las ventajas del SGML con un 20% de su complejidad".



Y es que los diseñadores de XML intentaron dejar fuera solo aquellas partes que raramente se utilizan. Esta reducción resultó ser muy importante: la especificación XML ocupa aproximadamente 30 páginas, frente a las 500 del SGML.

▸▸▸ PREPARÁNDONOS PARA TRABAJAR

Las herramientas y aplicaciones que son necesarias para poder trabajar con los ejemplos de esta guía son:.

▸ Un editor de textos, con el cual escribir los documentos XML y DTD.
▸ Un procesador o parser XML.
Más adelante, a medida que se vaya profundizando en el XML y tecnologías asociadas, irán haciendo falta nuevas aplicaciones que se describirán en su momento.

▸ **Editores XML:** Un editor XML es una aplicación que ofrece facilidades para crear y editar documentos XML. Para empezar, es recomendable que se utilice un simple editor de texto y, una vez que se esté familiarizado con la sintaxis y características del XML, se podrá pasar a editores que nos hagan la vida un poco más fácil (¡o más difícil!). Hay dos tipos de editores XML:

▸▸ Los que representan el archivo XML en forma de árbol, y permiten construir el documento trabajando sobre este árbol, y formularios adicionales. Algunos editores de este tipo son:

▸▸▸ XML Notepad: Desarrollado por Microsoft y para su utilización es necesario tener instalado, como mínimo, la versión 4.01 del Explorer, aunque solo se podrá aprovechar en su totalidad con la versión 5.

▸▸▸ Visual XML: Escrito en Java con JFC (Swing). Su autor es Pierre Morel.

▸▸ Los que presentan el documento XML en su formato original. Es decir, editores normales de archivos de texto, pero con facilidades de edición enfocadas al XML. Algunos de ellos son:

▸▸▸ XED: Desarrollado por Henry Thompson. Permite garantizar que el autor no va a escribir documentos que no estén bien formados, y puede leer la DTD para sugerir la introducción de elementos validos.

▸▸▸ PSGML para Emacs: Es un modo superior de Emacs para trabajar con SGML que se ha modificado para soportar XML. Lee DTD, puede utilizar un analizador externo para validar documentos, chequear la sintaxis, entre otras funciones.

En ambos tipos hay que diferenciar los que trabajan contra una DTD y, por lo tanto, validan el contenido de lo que se escribe; y los que simplemente aseguran que el documento XML está bien formado. Es decir, con la sintaxis correcta respecto de las especificaciones del XML.
La elección de uno u otro dependerá del tipo de documento que se esté escribiendo. Si se quiere escribir un documento XML que permita la introducción de datos, será preferible uno del primer tipo. Si se quiere escribir un documento con gran cantidad de texto (por ejemplo un manual), es preferible utilizar uno del segundo tipo. En cualquiera de los dos casos es recomendable que permita trabajar sobre una DTD.

▸▸▸ PARSERS XML

Un parser o procesador de XML es la herramienta principal de cualquier aplicación XML. Mediante este parser, no solo se puede comprobar si los documentos están bien formados y son válidos, sino que pueden además ser incorporados a las aplicaciones de manera que estas puedan manipular y trabajar con documentos XML.

▸ Actualmente hay muchos y para todos los lenguajes y plataformas: Java, C, Phyton, Visual Basic, Perl, Tcl, Delphi, etc., aunque los parsers en Java son la mayoría. Y es que Java y XML son la pareja perfecta al complementarse muy bien. El XML contribuye con datos independientes de la plataforma (documentos y datos portables). Java contribuye con el procesamiento independiente de la plataforma (soluciones de software portátiles orientadas a objetos). Como dijo Jon Bosak [1] en su conocido artículo, XML, Java y el futuro de la web: "XML dará trabajo a Java".

▸ Todas las grandes compañías ya han elaborado sus propios procesadores de XML, y existen muchos más completamente gratis. La utilización de uno u otro dependerá de nuestras necesidades, aunque es importante que siempre se tenga en cuenta la diferencia entre los que simplemente comprueban que el documento está bien formado, o el que valida respecto de un DTD.

▸ Si se posee una buena conexión a Internet, también pueden resultar útiles las siguientes direcciones, desde las que se puede validar un documento XML.

▸▸ RUWF *(http://www.xml.com/xml/pub/tools/ruwf/check.html)*, desde la cual se puede comprobar que un documento XML están bien formado. Simplemente hay que introducir la URL del documento XML.

▸▸ STG *(http://www.stg.brown.edu/service/xmlvalid/)*, desde donde se puede comprobar que un documento XML es válido. Se puede introducir la URL, escribirlo en la caja de texto de un formulario, o subir un archivo (upload) desde el disco duro.

EDITOR DTD

En una DTD se define cómo es la estructura de un documento XML; es decir, los elementos que formarán ese tipo de documento, y cómo están relacionados. Al contrario que en SGML, no son obligatorias en XML, aunque si es recomendable utilizarlas, al menos durante el periodo de diseño y validación de los documentos.
Para empezar, se puede utilizar un editor de textos habitual. Sin embargo, algunos específicos son:

▸ **EZDTD:** Además de permitir crear DTD de forma visual, tiene dos funcionalidades muy interesantes, como son el permitir guardar la DTD diferenciando si va a ser para documentos XML o SGML, y el permitir guardar la DTD en formato HTML, colocando enlaces de forma automática entre los elementos que define.

▸ **TDTD** para Emacs: Es un modo superior de Emacs para la edición de DTD. Chequea sintaxis, e incluye algunas macros que facilitan la edición de construcciones comunes.
También son interesantes las aplicaciones que dado un documento XML generan una DTD, como es el caso de:

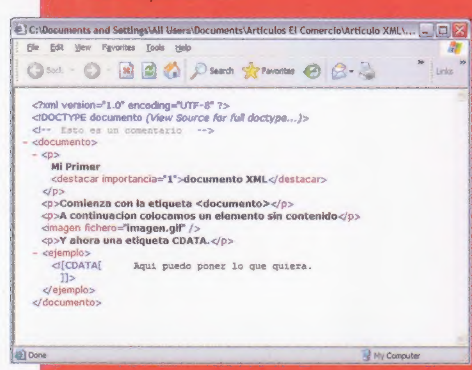
▸ **DTDGenerator:** Desde su web se puede subir un archivo XML, del cual generará una DTD.

EMPEZANDO A TRABAJAR CON XML

▸ Se va a estudiar con más detalle la sintaxis y los elementos que forman un documento XML, y cómo se puede construir y comprobar que es correcto. Por ejemplo, el siguiente código es un ejemplo de documento XML con DTD incorporada:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE documento [
<!ELEMENT documento (p | imagen | ejemplo)*>
<!ELEMENT p (#PCDATA|destacar)*>
<!ELEMENT destacar (#PCDATA)>
<!ATTLIST destacar
                importancia CDATA #REQUIRED>
<!ELEMENT imagen EMPTY>
<!ATTLIST imagen
                fichero CDATA #REQUIRED>
<!ELEMENT ejemplo (#PCDATA)>
]>
<!-- Esto es un comentario -->
<documento>
<p>Mi Primer <destacar importancia="1">documento XML</destacar></p>
<p>Comienza con la etiqueta &lt;documento&gt;</p>
<p>A continuación colocamos un elemento sin contenido</p>
<imagen fichero="imagen.gif"/>
<p>Y ahora una etiqueta CDATA.</p>
<ejemplo>
<![CDATA[
        Aqui puedo poner lo que quiera.
]]>
</ejemplo>
</documento>
```

▸ Para cualquier persona que esté familiarizada con el HTML, esta sintaxis le resultará conocida, aunque a simple vista se pueden observar algunas diferencias importantes:

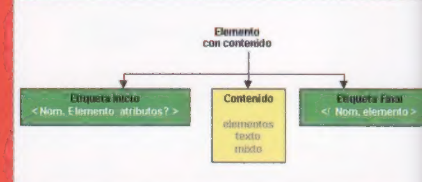


▸▸ Utilizo mis propias etiquetas: Y es que en XML no trabaja con etiquetas predefinidas. Puede crear su propio lenguaje de etiquetas en función de sus necesidades.
▸▸ La sintaxis es estricta: No se puede dejar de entrecomillar los atributos, o utilizar las mayúsculas y minúsculas sin ningún control. La especificación XML determina claramente una serie de reglas que especifican cuando un documento está bien formado.
▸▸ La utilización de una DTD: En HTML, a pesar de ser una aplicación SGML, no era obligatorio utilizarlas y aunque para trabajar con XML tampoco será necesario, sí es recomendable. Posiblemente no acompañen al documento XML en su distribución, pero resultan muy útiles en la elaboración y validación de los documentos.
▸▸ Los elementos vacíos: Son los elementos del tipo <img>, <hr>, etc. de HTML, en los que no existe etiqueta final al no tener contenido. Ahora, en el XML, la propia etiqueta de inicio llevará una contrabarra al final, que los identificará.

▸ **Marcado y datos**
Un documento XML es simplemente un conjunto de cadenas de caracteres [2], en el que, al igual que en el HTML, se pueden diferenciar dos tipos de construcciones: el marcado y los datos de caracter.
El texto incluido entre los caracteres menor que "<" y mayor que ">", o entre los signos "&" y ";", es el marcado. Son exactamente las partes del documento que tiene que entender el procesador de XML. El marcado entre los signos "<" y ">" se denominan etiqueta. El resto no son más que datos de caracter, que se corresponde con lo que sería el contenido del documento: es decir, la parte imprimible de éste.

▸ **Componentes de un documento XML**
▸▸ **Elementos:** Como se puede observar, todo documento XML se compone de uno o más elementos, cuyos límites están delimitados por etiquetas de comienzo y etiquetas de fin, en el caso de que tengan contenido (<p>Mi Primer <destacar importancia="1">documento XML</destacar></p>), y por una etiqueta de elemento vacío en el caso de ser elementos sin contenido (<imagen fichero="imagen.gif"/>).
Cada elemento puede contener datos de caracter, elementos, ambas cosas a la vez, o puede que estén vacíos.
En el ejemplo, el elemento "documento" está formado por otros elementos: "p", "imagen" y "ejemplo". El elemento "p" está formado por un contenido mixto: el elemento "destacar" y datos de carácter. El elemento "imagen" no tiene ningún contenido.

En el caso de elementos con contenido, las etiquetas de comienzo se componen del símbolo menor que "<", el nombre del tipo de elemento, los atributos si los tiene, y el símbolo mayor que ">". Mientras que las etiquetas de fin se componen del símbolo menor que seguido de contrabarra "</", el nombre del tipo del elemento y el símbolo mayor que ">".



En el caso de ser un elemento vacío, solo hay una etiqueta de elemento vacío que se forma del símbolo menor que "<", el nombre del tipo de ele-

Elemento vacío

↓

Etiqueta elemento vacio
< N. Elemento atributos? />

mento, los atributos si los tiene, y se cierra con el símbolo "/>". Es importante destacar este tipo de elementos, ya que hasta ahora en el SGML y, por tanto en el HTML, entendido como aplicación SGML; los elementos vacíos solo se representaban con una etiqueta de inicio.

►► **Atributos:** Cada elemento puede tener atributos (propiedades) que ofrecen información sobre el elemento. En el ejemplo, el elemento, "destacar" va caracterizado con el atributo "importancia", que indicará el grado de relevancia de su contenido. El elemento "imagen" con el atributo "fichero", donde indicará el archivo que contiene la imagen.

```
<p>Mi Primer <destacar importancia="1">documento XML</destacar></p>
.....
<imagen fichero="imagen.gif"/>
```

Como se puede observar, la definición de un atributo está formada por el nombre del atributo, seguido del símbolo igual "=" y, entrecomillado, el valor del atributo.

►► **Prólogo:** Los documentos XML pueden empezar con un prólogo, en el que esencialmente se define, una declaración XML y una declaración de tipo de documento

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE documento [
<!ELEMENT documento (p | imagen | ejemplo)*>
<!ELEMENT p (#PCDATA|destacar)*>
<!ELEMENT destacar (#PCDATA)>
<!ATTLIST destacar
            importancia CDATA #REQUIRED>
<!ELEMENT imagen EMPTY>
<!ATTLIST imagen
            fichero CDATA #REQUIRED>
<!ELEMENT ejemplo (#PCDATA)>
]>
```

►►► En la declaración XML (<?xml version="1.0" encoding="UTF-8"?>) se indica, información sobre la versión de XML que se está utilizando, e información sobre el tipo de codificación de caracteres que se está usando. En este caso es el código ASCII de 7 bits, que es un subconjunto del código Unicode denominado UTF-8. No hubiese sido necesario declararlo, ya que es el que los parsers manejan por defecto.

►►► En la declaración del tipo de documento:

```
<!DOCTYPE documento [
<!ELEMENT documento (p | imagen | ejemplo)*>
<!ELEMENT p (#PCDATA|destacar)*>
<!ELEMENT destacar (#PCDATA)>
<!ATTLIST destacar
            importancia CDATA #REQUIRED>
<!ELEMENT imagen EMPTY>
<!ATTLIST imagen
            fichero CDATA #REQUIRED>
<!ELEMENT ejemplo (#PCDATA)>
]>
```

Se asocia el DTD respecto de la cual se construye el documento. En el ejemplo va implícita en el propio documento XML, aunque también puede hacerse externa al documento, e incluso de una forma mixta. Si se hubiese escrito en un archivo "ejemplo.dtd", se indicaría de la siguiente manera: <!DOCTYPE documento SYSTEM "ejemplo.dtd">. Aunque cabe recordar que a diferencia del SGML, se tiene la posibilidad de no utilizarla. Ambas partes del prólogo son opcionales, aunque en el caso de incluir ambas, la declaración XML tiene que ir antes.

►► **Comentarios:** Mediante los cuales se proporciona información que el parser no tendrá en cuenta.

```
<!-- Esto es un comentario -->
```

Los comentarios empiezan con los caracteres "<!--" y terminan con "-->", y pueden colocarse en cualquier sitio excepto dentro de las declaraciones, etiquetas y otros comentarios.

►► **CDATA:** Permiten integrar texto en un documento XML, que de otra forma sería interpretado como etiquetas. Es decir, se introduce texto que luego el procesador XML va a mostrar, pero no va a procesar como marcado.

```
<![CDATA[
        Aquí puedo poner lo que quiera.
]]>
```

Los CDATA empiezan con los caracteres "<![CDATA[" y termina con "]]>". Dentro de ellos se puede colocar cualquier cosa, ya que no va a ser interpretado, con la salvedad de la cadena que indica el final de CDATA, "]]>".

►► **Entidades predefinidas:** En XML existen algunos caracteres reservados que no se pueden utilizar para evitar problemas con el marcado. Las entidades predefinidas son marcas XML que se utilizan para representar estos caracteres. El XML especifica cinco entidades predefinidas:

- ►► & para el &
- ►► < para el <
- ►► > para el >
- ►► ' para el '
- ►► " para el "

► **Documentos bien formados y documentos válidos:** Como ya se ha comentado, no es necesario que un documento XML esté asociado a una DTD. El documento XML expresado al inicio de esta ficha se podría haber escrito de la siguiente manera, y si se pasa por un parser de XML no daría ningún error:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Esto es un comentario -->
<documento>
<p>Mi Primer <destacar importancia="1">documento XML</destacar></p>
<p>Comienza con la etiqueta &lt;documento&gt;</p>
<p>A continuacion colocamos un elemento sin contenido</p>
<imagen fichero="imagen.gif"/>
<p>Y ahora una etiqueta CDATA.</p>
<ejemplo>
<![CDATA[
        Aqui puedo poner lo que quiera.
]]>
</ejemplo>
</documento>
```

Por tanto, en función de si lleva asociada una DTD o no, se puede diferenciar dos tipos de documentos XML: Válidos (aquellos que siguen las reglas de una DTD especifica), y bien formados (well-formed, que no tienen necesariamente una DTD asociada, pero siguen las reglas del XML al pie de la letra). Evidentemente, los documentos válidos son bien formados.

DOCUMENTOS XML BIEN FORMADOS

Se ha visto de forma superficial que una de las características que diferencian al XML del SGML es la posibilidad de no utilizar DTD. Sin embargo, en una DTD se define cómo va a ser un tipo de documento (los elementos, atributos y entidades que lo van a formar, cómo se estructuran y relacionan). Por tanto, si en la elaboración de un documento XML no se utiliza una DTD, el parser no puede proporcionar información sobre la validez de ese documento; es decir, no nos puede indicar que los elementos y atributos que se utilizan son los correctos y que se encuentran en el orden adecuado. Simplemente indicará si ese documento está bien formado; es decir, si respeta las reglas sintácticas del lenguaje XML.
Según la especificación, un objeto de texto es un documento XML bien formado si:

- Tomado como un todo, cumple la regla denominada "document".
- Respeta todas las restricciones de buena formación dadas en la especificación.
- Cada una de las entidades analizadas que se referencia directa o indirectamente en el documento está bien formada.

La regla "document": Cumplir la regla "document" significa:

- Que contiene uno o más elementos.
- Hay exactamente un elemento, llamado raíz, o elemento documento, del cual ninguna parte aparece en el contenido de ningún otro elemento.
- Para el resto de elementos, si la etiqueta de comienzo está en el contenido de algún otro elemento, la etiqueta de fin está en el contenido del mismo elemento. Es decir, los elementos delimitados por etiquetas de principio y final, se anidan adecuada y mutuamente.

El siguiente ejemplo no es un documento XML bien formado:

```
Mi primer documento XML
```

ya que no contiene ningún elemento y, por tanto, está incumpliendo la regla número 1.
En cambio:

```
<p>Mi primer documento XML</p>
```

Sí lo es, al contener al menos el elemento "p". La principal razón por la que el procesador comprueba los elementos es para determinar si el documento tiene estructura de datos que pueda extraer. Un documento que carece de elementos no tiene estructura de datos. Un documento con al menos un elemento tiene estructura de datos.
En cambio:

```
<p>Mi primer documento XML</p>
<p>Mi primer documento XML</p>
```

No es un documento XML bien formado al incumplir la regla número 2, según la cual solo puede existir un único elemento raíz.
Aunque escrito de la siguiente manera si que es correcto:

```
<documento>
        <p>Mi primer documento XML</p>
        <p>Mi primer documento XML</p>
</documento>
```

Al convertirse el elemento "documento" en el elemento raíz, ser único, y no formar parte del contenido de ningún otro elemento.
En cambio, el siguiente ejemplo:

```
<documento>
        <p>Mi primer <destacar>documento XML</p
></destacar>
        <p>Mi primer documento XML</p>
</documento>
```

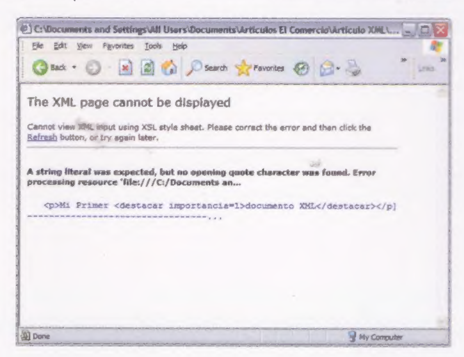
Es incorrecto al incumplir la regla 3, ya que la etiqueta inicio del elemento "destacar" está dentro del contenido del elemento "p", pero su etiqueta final está fuera.
La forma correcta sería la siguiente:

```
<documento>
        <p>Mi primer <destacar>documento
XML </destacar></p>
        <p>Mi primer documento XML</p>
</documento>
```

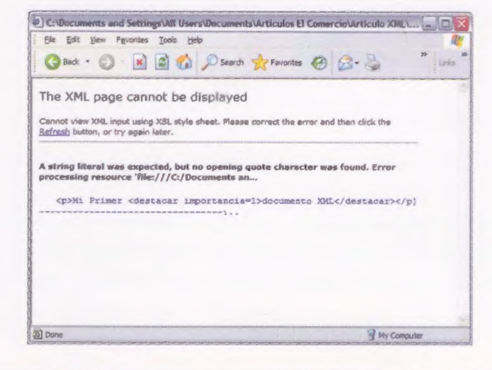
Sintaxis correcta y restricciones de buena formación: Además de las reglas anteriormente mencionadas, para escribir documentos XML bien formados hay que conocer perfectamente la sintaxis del lenguaje XML, y algunas restricciones que la especificación impone. Algunas de estas reglas ya se han visto anteriormente: cómo se escriben las etiquetas de inicio y final, cómo se escriben las etiquetas de elementos vacíos, cómo se escriben los atributos, etc., y es evidente que si se siguen estas reglas, el parser dará error. En el siguiente ejemplo se presentan cuatro errores:

```
<?xml version="1.0"?>
<documento>
 <p>Mi Primer <destacar importancia=1>documento
XML
</destacar></p]
 <p>Comienza con la etiqueta <documento&gt;</p>
 <p>A continuacion colocamos un elemento sin conteni
do</p>
 <imagen fichero="imagen.gif">
</documento>
```

- El valor del atributo "importancia", no está entrecomillado. En HTML es posible no entrecomillar el valor de los atributos, pero en XML es obligatorio. Se debería haber escrito: ...<destacar importancia="1">documento XML</destacar>...
- La etiqueta final, del elemento "p" está mal cerrada. En lugar del carácter "]" , se debería poner el símbolo mayor que ">". <p>Mi Primer <destacar importancia=1>documento XML</destacar></p>



- Se está utilizando el símbolo menor que "<" sin que forme parte de la definición de una etiqueta. Al ser un carácter reservado, habría que escribirlo como la entidad predefinida <. <p>Comienza con la etiqueta <documento></p>
- Se está escribiendo el elemento vacío "imagen" de forma incorrecta. Al ser un elemento sin contenido, habría que haberlo escrito con una etiqueta de elemento vacío: <imagen fichero="imagen.gif"/>. O también de la siguiente manera: <imagen fichero="imagen.gif"></imagen>



DOCUMENTOS XML BIEN FORMADOS

El ejemplo anterior bien escrito sería:

```
<?xml version="1.0"?>
<documento>
  <p>Mi Primer <destacar importancia="1">documento XML</destacar></p>
  <p>Comienza con la etiqueta &lt;documento&gt;</p>
  <p>A continuacion colocamos un elemento sin contenido</p>
  <imagen fichero="imagen.gif"/>
</documento>
```

▸ **Otras reglas que se deben tener en cuenta son:**

▸▸ El XML es sensible a la utilización de mayúsculas y minúsculas. En el siguiente ejemplo:

```
<p>Mi primer documento XML</p>
<P>Mi primer documento XML</P>
```

Los elementos "p" y "P" son diferentes. Hay que tener mucho cuidado con esta regla, ya que es habitual su incumplimiento, y suele ser la causa de la mayor parte de los errores. Es recomendable establecer un criterio al respecto, antes de empezar a escribir un documento.

El nombre de la etiqueta de inicio y final debe ser el mismo. El siguiente ejemplo es incorrecto: <p>Mi primer documento XML</P>, ya que al hacer diferencia entre mayúsculas y minúsculas, el parser no entiende ambas etiquetas como del mismo elemento.

Ningún nombre de atributo puede aparecer más de una vez en la misma etiqueta de inicio, o de elemento vacío. El siguiente ejemplo es incorrecto: **...<destacar importancia="1" importancia="2">documento XML</destacar>...**

▸▸ **Las entidades:** Todavía no se ha hablado mucho de las entidades pero, como se verá, resultan muy importantes en la elaboración y mantenimiento eficiente de documentos XML. Una de sus funcionalidades es la de permitir elaborar un documento XML en "trozos"; es decir, se tendrá un único documento XML, pero éste físicamente se encontrará dividido en varios archivos.

No se va a profundizar más en este aspecto, pero lo que hay que tener en cuenta es que para el parser se trata de un único documento XML y que, por tanto, sus diferentes partes, aún encontrándose en archivos diferentes, deben verificar las reglas de buena formación descritas anteriormente.

SCHEMAS XML

▸ Un "schema XML" es algo similar a un DTD; es decir, define qué elementos puede contener un documento XML, cómo están organizados, que atributos y de qué tipo pueden ser sus elementos. La ventaja de los schemas con respecto a los DTD son:

▸▸ Usan sintaxis de XML, al contrario que los DTD.

▸▸ Permiten especificar los tipos de datos.

▸▸ Son extensibles.

▸ Por ejemplo, un schema permite definir el tipo del contenido de un elemento o de un atributo, y especificar si debe ser un número entero, o una cadena de texto, o una fecha, etc. Los DTD no permiten hacer estas cosas.

Veamos un ejemplo de un documento XML, y su schema correspondiente:

```
<documento xmlns="x-schema:personaSchema.xml">
<persona id="fulano">
<nombre>Fulano Menganez</nombre>
</persona>
</documento>
```

▸ Como podemos ver en el documento XML anterior, se hace referencia a un espacio de nombres (namespace) llamado "x-schema:personaSchema.xml". Es decir, le estamos diciendo al analizador sintáctico XML (parser) que valide el documento contra el schema "personaSchema.xml".

El schema sería algo parecido a esto:

```
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
<AttributeType name='id' dt:type='string' required='yes'/>
<ElementType name='nombre' content='textOnly'/>
<ElementType name='persona' content='mixed'>
<attrubyte type='id'/>
<element type='nombre'/>
</ElementType>
<ElementType name='documento' content='eltOnly'>
<element type='persona'/>
</ElementType>
</Schema>
```

▸ El primer elemento del schema define dos espacios de nombre. El primero, "xml-data", le dice al analizador que esto es un schema, y no otro documento XML cualquiera. El segundo, "datatypes", permite definir el tipo de elementos y atributos utilizando el prefijo "dt".

▸ ElementType: Define el tipo y contenido de un elemento, incluyendo los sub-elementos que pueda contener.

▸ AttributeType: Asigna un tipo y condiciones a un atributo.

▸ attribute: Declara que un atributo previamente definido por AttributeType, puede aparecer como atributo de un elemento determinado.

▸ element: Declara que un elemento previamente definido por ElementType puede aparecer como contenido de otro elemento.

▸ Tal como hemos visto, es necesario empezar el schema definiendo los elementos más profundamente anidados dentro de la estructura jerárquica de elementos del documento XML. Es decir, tenemos que trabajar "de dentro hacia fuera".

▸ Visto de otra manera, las declaraciones de tipo ElementType y AttributeType deben preceder a las declaraciones de contenido element y attribute correspondientes.

ANOTACIONES

[1] Jon Bosak es el padre del XML. No solo fue el que tuvo las ideas originales y seleccionó el equipo de trabajo, sino que organiza y dirige actualmente el grupo de trabajo de XML del W3C. Sin Bosak, el XML no hubiera existido.

[2] Procedentes del conjunto de caracteres Unicode. Es el juego internacional de caracteres estándar y que utiliza códigos de 16 bits. De esta forma es posible disponer de una tabla con 65.535 símbolos disponibles, lo que permite acomodar juegos de caracteres de las diversas lenguas sin mayores problemas.

▸ **DIRECTOR:** BERNARDO ROCA REY MIRÓ QUESADA ▸ **EDITORA:** FRANCA CAVASSA ▸ **EDITORA ADJUNTA:** MELISA OSORES ▸ **DESARROLLO:** HERNÁN BALDERRAMA / LUIS FIGALA ▸ **GERENTE COMERCIAL:** AUGUSTO TOWSEND ▸ **GERENTE DE PRODUCTO:** RENZO MARIÁTEGUI ▸ **JEFE DE PRODUCTO:** PAOLA CORNEJO ▸ **COORDINACIÓN:** DORA NIQUÉN ▸ **EDITORA DE DISEÑO:** CLAUDIA BURGA-CISNEROS ▸ **DISEÑO:** VERUZKA NORIEGA ▸ **DIAGRAMACIÓN:** GERARDO CRISTÓBAL ▸ **EDICIÓN, PREPRENSA E IMPRESIÓN:** EMPRESA EDITORA EL COMERCIO S.A., PERÚ, AÑO 2004 TODOS LOS DERECHOS RESERVADOS. EMPRESA EDITORA EL COMERCIO S.A.